Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 00-017

Centroid-Based Document Classification Algorithms: Analysis & Experimental Results

Euihong (sam) Han and George Karypis

March 06, 2000

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **06 MAR 2000**	2. REPORT TYPE	3. DATES COVERED **-**
4. TITLE AND SUBTITLE **Centroid-Based Document Classification Algorithms: Analysis & Experimental Results**		5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Army Research Office,PO Box 12211,Research Triangle Park,NC,27709-2211**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**17**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Centroid-Based Document Classification: Analysis & Experimental Results*

Eui-Hong (Sam) Han and George Karypis

University of Minnesota, Department of Computer Science / Army HPC Research Center
Minneapolis, MN 55455
Technical Report: #00-017

{han, karypis}@cs.umn.edu

Last updated on March 6, 2000 at 12:27am

Abstract

In recent years we have seen a tremendous growth in the volume of text documents available on the Internet, digital libraries, news sources, and company-wide intranets. Automatic text categorization, which is the task of assigning text documents to pre-specified classes (topics or themes) of documents, is an important task that can help both in organizing as well as in finding information on these huge resources. Text categorization presents unique challenges due to the large number of attributes present in the data set, large number of training samples, and attribute dependencies. In this paper we focus on a simple linear-time centroid-based document classification algorithm, that despite its simplicity and robust performance, has not been extensively studied and analyzed. Our extensive experiments show that this centroid-based classifier consistently and substantially outperforms other algorithms such as Naive Bayesian, *k*-nearest-neighbors, and C4.5, on a wide range of datasets. Our analysis shows that the similarity measure used by the centroid-based scheme allows it to classify a new document based on how closely its behavior matches the behavior of the documents belonging to different classes, as measured by the average similarity between the documents. This matching allows it to dynamically adjust for classes with different densities. Furthermore, our analysis shows that the similarity measure of the centroid-based scheme accounts for dependencies between the terms in the different classes. We believe that this feature is the reason why it consistently outperforms other classifiers that cannot take these dependencies into account.

1 Introduction

We have seen a tremendous growth in the volume of online text documents available on the Internet, digital libraries, news sources, and company-wide intranets. It has been forecasted that these documents (with other unstructured data) will become the predominant data type stored online [40]. This provides a huge opportunity to make more

*This work was supported by NSF CCR-9972519, by Army Research Office contract DA/DAAG55-98-1-0441, by the DOE ASCI program, and by Army High Performance Computing Research Center contract number DAAH04-95-C-0008. Access to computing facilities was provided by AHPCRC, Minnesota Supercomputer Institute. Related papers are available via WWW at URL: **http://www.cs.umn.edu/˜karypis**

effective use of these collections and there is a growing need for tools to deal with text documents. Automatic text categorization [54, 56, 55, 43, 41, 34, 6, 22], which is the task of assigning text documents to pre-specified classes (topics or themes) of documents, is an important task that can help people to find information on these huge resources.

Text categorization presents unique challenges due to the large number of attributes present in the data set, large number of training samples, attribute dependency, and multi-modality of categories. This has led to the development of a variety of text categorization algorithms [31, 22, 25, 2, 55, 26, 19] that address these challenges to varying degrees. In this paper we focus on a simple centroid-based document classification algorithm that has not been extensively studied and analyzed despite its simplicity and, as our experiments show, its robust performance.

In this algorithm, a centroid vector is computed to represent the documents of each class, and a new document is assigned to the class that corresponds to its most similar centroid vector, as measured by the cosine function. The computational complexity of the learning phase of this algorithm is linear on the number of documents, and for each new document, its classification complexity is linear on the number of classes. Extensive experiments presented in Section 4 show that this centroid-based classifier consistently and substantially outperforms other algorithms such as Naive Bayesian, k-nearest-neighbors, and C4.5, on a wide range of datasets. The surprisingly good classification performance of this scheme suggests that it utilizes a powerful classification model.

In this paper we present such an analysis. Our analysis shows that the similarity measure used by the centroid-based scheme allows it to classify a new document based on how closely its behavior matches the behavior of the documents belonging to different classes, as measured by the average similarity between the documents. This matching allows it to dynamically adjust for classes with different densities. Our analysis also shows that the similarity measure of the centroid-based scheme can account for dependencies between the terms in the different classes. We believe that this feature of the centroid-based classifier is the reason why it consistently outperforms the Naive Bayesian classifier, which can not take these dependencies into account.

The reminder of the paper is organized as follows. Section 2 provides an overview of some of the algorithms that have been used for document categorization. Section 3 describes the centroid-based document classification algorithm. Section 4 experimentally evaluates this algorithm on a variety of data sets. Section 5 analyzes the classification model of the centroid-based classifier and compares it against those used by other algorithms. Finally, Section 6 provides directions for future research.

2 Previous Work

The various document categorization algorithms that have been developed over the years [47, 1, 10, 17, 31, 22, 25, 2, 55, 26, 19] fall under two general categories. The first category contains traditional machine learning algorithms such as decision trees, rule sets, instance-based classifiers, probabilistic classifiers, support vector machines, *etc*., that have either been used directly or being adapted for use in the context of document data sets. On the other hand, the second category contains specialized categorization algorithms developed in the Information Retrieval community. Examples of such algorithms include relevance feedback, linear classifiers, generalized instance set classifiers, *etc*. In the rest of this section we briefly describe some of these algorithms and discuss their merits for document categorization.

k Nearest Neighbor k-nearest neighbor (k-NN) classification is an instance-based learning algorithm that has been applied to text categorization since the early days of research [33, 21, 52, 6], and has been shown to produce better results when compared against other machine learning algorithms such as C4.5 [39] and RIPPER [5]. In this classification paradigm, k nearest neighbors of a test document are computed first. Then the similarities of this document to the k nearest neighbors are aggregated according to the class of the neighbors, and the test document is assigned to the most similar class (as measured by the aggregate similarity). A major drawback of the similarity measure used in k-NN is that it uses all features equally in computing similarities. This can lead to poor similarity

measures and classification errors, when only a small subset of the words is useful for classification. To address this problem, a variety of techniques have been developed for adjusting the importance of the various terms in a supervised setting. Examples of such techniques include preset weight adjustment using mutual information [11, 49, 48], RELIEF [23, 24], and variable-kernel similarity metric learning [32].

C4.5 A decision tree is a widely used classification paradigm in machine learning and data mining. The decision tree model is built by recursively splitting the training set based on a locally optimal criterion until all or most of the records belonging to each of the leaf nodes bear the same class label. C4.5 [39] is a widely used decision tree-based classification algorithm that has been shown to produce good classification results, primarily on low dimensional data sets. Unfortunately, one of the characteristics of document data sets is that there is a relatively large number of features that characterize each class. Decision tree based schemes like C4.5 do not work very well in this scenario due to overfitting [6, 19]. The overfitting occurs because the number of samples is relatively small with respect to the number of distinguishing words, which leads to very large trees with limited generalization ability.

Naive Bayesian The naive Bayesian (NB) algorithm has been widely used for document classification, and has been shown to produce very good performance [28, 29, 27, 34]. For each document, the naive Bayesian algorithm computes the posterior probability that the document belongs to different classes and assigns it to the class with the highest posterior probability. The posterior probability $P(c_k|d_i)$ of class c_k given a test document d_i is computed using Bayes rule

$$P(c_k|d_i) \quad = \quad \frac{P(c_k)P(d_i|c_k)}{P(d_i)}, \tag{1}$$

and d_i is assigned to the class with the highest posterior probability, that is,

$$\text{Class of } d_i = \arg\max_{1\le k\le N}\{P(c_k|d_i)\} = \arg\max_{1\le k\le N}\{P(c_k)P(d_i|c_k)\}, \tag{2}$$

where N is the total number of classes.

The naive Bayesian algorithm models each document d_i, as a vector in the term space, *i.e.*, $d_i = (d_{i1}, d_{i2}, \ldots, d_{im})$, where d_{ij} models the presence or absence of the jth term. Naive Bayesian computes the two quantities required in Equation 2 as follows. The approximate class priors ($P(c_k)$) are computed using the maximum likelihood estimate

$$P(c_k) \quad = \quad \frac{\sum_{i=1}^{|D|} P(c_k|d_i)}{|D|}, \tag{3}$$

where D is the set of training documents and $|D|$ is the number of training documents in D. The $P(d_i|c_k)$ is computed by assuming that when conditioned on a particular class c_k, the occurrence of a particular value of d_{ij} is statistically independent of the occurrence of any other value in any other term $d_{ij'}$. Under this assumption, we have that

$$P(d_i|c_k) \quad = \quad \prod_{j=1}^{m} P(d_{ij}|c_k), \tag{4}$$

and because of this assumption this classifier is called "naive" Bayesian. The computation of $P(d_{ij}|c_k)$ in Equation 4 varies according to the model chosen for document representation. There are two popular models for representing documents [34]. The first is the multi-variate Bernoulli event model that only takes into account the presence or absence of a particular term, and does not account for term frequency. The second model is the multinomial model that captures the word frequency information.

Despite the fact that the independence assumption of naive Bayesian does not hold in real document data sets,

Naive-Bayesian classifiers perform surprisingly well [54, 56, 27, 34], in practice. Domingos and Pazzani [14] provide an explanation for the relatively good performance of Naive-Bayesian classifiers [14]. They argue that even though Naive-Bayesian classifiers do not estimate the underlying probability densities correctly, they provide good enough solutions in terms of zero-one loss (misclassification rate).

Linear Classifiers Linear classifiers [31] are a family of text categorization learning algorithms that learn a feature weight vector for every category. The weight learning techniques such as Rocchio [42] and Widrow-Hoff algorithm [50] are used to learn the feature weight vector from the training samples. These weight learning algorithms adjust the feature weight vector such that features or words that contribute significantly to the categorization have large values. A test document is determined to belong to a particular category if the dot product between the test document and the feature weight vector is greater than a certain threshold value.

Generalized Instance Set Algorithm Generalized Instance Set (GIS) Algorithm [25] is a text categorization algorithm that combines the advantage of kNN and linear classifiers. The feature weight vector of a category in linear classifiers can be regarded as single generalized instance of the category. This feature weight vector in effect summarizes the entire category. In GIS, multiple generalized instances are found per category. Each generalized instance is a feature weight vector that is learned from the set of similar training samples. A test document is classified according to the sum of similarities to these generalized instances. GIS inherits expressive power of kNN by having multiple feature weight vectors per category and avoids the problem of kNN by learning feature weights using the weight learning techniques of linear classifiers.

Support Vector Machines Support Vector Machines (SVM) is a new learning algorithm proposed by Vapnik [46]. This algorithm was introduced to solve two-class pattern recognition problem using the Structural Risk Minimization principle [46, 7]. Given a training set in a vector space, this method finds the *best* decision hyperplane that separates two classes. The quality of a decision hyperplane is determined by the distance (referred as margin) between two hyperplanes that are parallel to the decision hyperplane and touch the closest data points of each class. The *best* decision hyperplane is the one with the maximum margin. The SVM problem can be solved using quadratic programming techniques [46, 7]. SVM extends its applicability on the linearly non-separable data sets by either using soft margin hyperplanes, or by mapping the original data vectors into a higher dimensional space in which the data points are linearly separable. An efficient implementation of SVM and its application in text categorization of Reuters-21578 corpus is reported in [22].

3 Centroid-Based Document Classifier

In the centroid-based classification algorithm, the documents are represented using the vector-space model [43]. In this model, each document d is considered to be a vector in the term-space. In its simplest form, each document is represented by the *term-frequency* (TF) vector $\vec{d}_{tf} = (tf_1, tf_2, \ldots, tf_n)$, where tf_i is the frequency of the ith term in the document. A widely used refinement to this model is to weight each term based on its *inverse document frequency* (IDF) in the document collection. The motivation behind this weighting is that terms appearing frequently in many documents have limited discrimination power, and for this reason they need to be de-emphasized. This is commonly done [43] by multiplying the frequency of each term i by $\log(N/df_i)$, where N is the total number of documents in the collection, and df_i is the number of documents that contain the ith term (*i.e.*, document frequency). This leads to the *tf-idf* representation of the document, *i.e.*, $\vec{d}_{tfidf} = (tf_1 \log(N/df_1), tf_2 \log(N/df_2), \ldots, tf_n \log(N/df_n))$. Finally, in order to account for documents of different lengths, the length of each document vector is normalized so that it is of unit length, *i.e.*, $\|\vec{d}_{tfidf}\|_2 = 1$. In the rest of the paper, we will assume that the vector representation $\vec{d}$ of each document d has been weighted using *tf-idf* and it has been normalized so that it is of unit length.

In the vector-space model, the similarity between two documents d_i and d_j is commonly measured using the cosine function [43], given by

$$\cos(\vec{d_i}, \vec{d_j}) = \frac{\vec{d_i} \cdot \vec{d_j}}{\|\vec{d_i}\|_2 * \|\vec{d_j}\|_2}, \tag{5}$$

where "·" denotes the dot-product of the two vectors. Since the document vectors are of unit length, the above formula simplifies to $\cos(\vec{d_i}, \vec{d_j}) = \vec{d_i} \cdot \vec{d_j}$.

Given a set S of documents and their corresponding vector representations, we define the ***centroid*** vector $\vec{C}$ to be

$$\vec{C} = \frac{1}{|S|} \sum_{d \in S} \vec{d}, \tag{6}$$

which is nothing more than the vector obtained by averaging the weights of the various terms present in the documents of S. We will refer to the S as the ***supporting set*** for the centroid $\vec{C}$. Analogously to documents, the similarity between two centroid vectors and between a document and a centroid vector are computed using the cosine measure. In the first case,

$$\cos(\vec{C_i}, \vec{C_j}) = \frac{\vec{C_i} \cdot \vec{C_j}}{\|\vec{C_i}\|_2 * \|\vec{C_j}\|_2}, \tag{7}$$

whereas in the second case,

$$\cos(\vec{d}, \vec{C}) = \frac{\vec{d} \cdot \vec{C}}{\|\vec{d}\|_2 * \|\vec{C}\|_2} = \frac{\vec{d} \cdot \vec{C}}{\|\vec{C}\|_2}. \tag{8}$$

Note that even though the document vectors are of length one, the centroid vectors will not necessarily be of unit length.

The idea behind the centroid-based classification algorithm is extremely simple. For each set of documents belonging to the same class, we compute their centroid vectors. If there are k classes in the training set, this leads to k centroid vectors $\{\vec{C_1}, \vec{C_2}, \ldots, \vec{C_k}\}$, where each $\vec{C_i}$ is the centroid for the ith class. The class of a new document x is determined as follows. First we use the document-frequencies of the various terms computed from the training set to compute the *tf-idf* weighted vector-space representation of x, and scale it so $\vec{x}$ is of unit length. Then, we compute the similarity between $\vec{x}$ to all k centroids using the cosine measure. Finally, based on these similarities, we assign x to the class corresponding to the most similar centroid. That is, the class of x is given by

$$\arg \max_{j=1,\ldots,k} (\cos(\vec{x}, \vec{C_j})). \tag{9}$$

The computational complexity of the learning phase of this centroid-based classifier is linear on the number of documents and the number of terms in the training set. The computation of the vector-space representation of the documents can be easily computed by performing at most three passes through the training set. Similarly, all k centroids can be computed in a single pass through the training set, as each centroid is computed by averaging the documents of the corresponding class. Moreover, the amount of time required to classify a new document x is at most $O(km)$, where m is the number of terms present in x. Thus, the overall computational complexity of this algorithm is very low, and is identical to fast document classifiers such as Naive Bayesian.

4 Experimental Results

We evaluated the performance of the centroid-based classifier by comparing against the naive Bayesian, C4.5, and k-nearest-neighbor classifiers on a variety of document collections. We obtained the naive Bayesian results using the Rainbow [35] software library. Rainbow is a state-of-art implementation of the Naive Bayesian algorithm for text classification [34]. Rainbow has options for both the multi-variate Bernoulli event model and the multinomial

event model. Experiments reported in [34] show that the multinomial event model works better than the multi-variate Bernoulli event model, and this is the model used in our experiments. The C4.5 results were obtained using a locally modified version of the C4.5 algorithm capable of handling sparse data sets. Finally, the k-nearest-neighbor results were obtained by using the *tf-idf* vector-space representation of the documents (identical to that used by the centroid-based classification algorithm), we used $k = 10$.

4.1 Document Collections

Data	Source	# of doc	# of class	min class size	max class size	avg class size	# of words
west1	West Group	500	10	39	73	50.0	977
west2	West Group	300	10	18	45	30.0	1078
west3	West Group	245	10	17	34	24.5	1035
oh0	OHSUMED-233445	1003	10	51	194	100.3	3182
oh5	OHSUMED-233445	918	10	59	149	91.8	3012
oh10	OHSUMED-233445	1050	10	52	165	105.0	3238
oh15	OHSUMED-233445	913	10	53	157	91.3	3100
ohscal	OHSUMED-233445	11162	10	709	1621	1116.2	11465
re0	Reuters-21578	1504	13	11	608	115.7	2886
re1	Reuters-21578	1657	25	10	371	66.3	3758
tr11	TREC	414	9	6	132	46.0	6429
tr12	TREC	313	8	9	93	39.1	5804
tr21	TREC	336	6	4	231	56.0	7902
tr23	TREC	204	6	6	91	34.0	5832
tr31	TREC	927	7	2	352	132.4	10128
tr41	TREC	878	10	9	243	87.8	7454
tr45	TREC	690	10	14	160	69.0	8261
la1	TREC	3204	6	273	943	534.0	31472
la2	TREC	3075	6	248	905	512.5	31472
la12	TREC	6279	6	521	1848	1046.5	31472
fbis	TREC	2463	17	38	506	144.9	2000
new3	TREC	9558	44	104	696	217.2	83487
wap	WebACE	1560	20	5	341	78.0	8460

Table 1: Summary of data sets used.

The characteristics of the various document collections used in our experiments are summarized in Table 1. The first three data sets are from the statutory collections of the legal document publishing division of West Group described in [8]. Data sets *tr11*, *tr12*, *tr21*, *tr23*, *tr31*, *tr41*, *tr45*, and *new3* are derived from TREC-5 [45], TREC-6 [45], and TREC-7 [45] collections. Data set *fbis* is from the Foreign Broadcast Information Service data of TREC-5 [45]. Data sets *la1*, *la2*, and *la12* are from the Los Angeles Times data of TREC-5 [45]. The classes of the various *trXX*, *new3*, and *fbis* data sets were generated from the relevance judgment provided in these collections. The class labels of *la1*, *la2*, and *la12* were generated according to the name of the news-paper sections that these articles appeared, such as “Entertainment”, “Financial”, “Foreign”, “Metro”, “National”, and “Sports”. Data sets *re0* and *re1* are from Reuters-21578 text categorization test collection Distribution 1.0 [30]. We divided the labels into 2 sets and constructed data sets accordingly. For each data set, we selected documents that have a single label. Data sets *oh0*, *oh5*, *oh10*, *oh15*, and *ohscal* are from OHSUMED collection [20] subset of MEDLINE database, which contains 233,445 documents indexed using 14,321 unique categories. We took different subsets of categories to construct these data sets. Data set *wap* is from the WebACE project (WAP) [37, 18, 3, 4]. Each document corresponds to a web page listed in the subject hierarchy of Yahoo! [51]. For all data sets, we used a stop-list to remove common words, and the words were stemmed using Porter’s suffix-stripping algorithm [38].

4.2 Classification Performance

The classification accuracy of the various algorithms on the different data sets in our experimental testbed are shown in Table 2. These results correspond to the average classification accuracies of 10 experiments. In each experiment 80% of the documents were randomly selected as the training set, and the remaining 20% as the test set. The first three columns of this table, show the results for the naive Bayesian, C4.5, and k-nearest neighbor schemes, whereas the last column shows the results achieved by the centroid-based classification algorithm (denoted as "Cntr" in the table). For each one of the data sets, we used a boldface font to highlight the algorithm that achieved the highest classification accuracy.

	NB	C4.5	kNN	Cntr
west1	86.7	85.5	82.9	**87.5**
west2	76.5	75.3	77.2	**79.0**
west3	75.1	73.5	76.1	**81.6**
oh0	89.1	82.8	84.4	**89.3**
oh5	87.1	79.6	85.6	**88.2**
oh10	81.2	73.1	77.5	**85.3**
oh15	84.0	75.2	81.7	**87.4**
re0	**81.1**	75.8	77.9	79.8
re1	**80.5**	77.9	78.9	80.4
tr11	85.3	78.2	85.3	**88.2**
tr12	79.8	79.2	85.7	**90.3**
tr21	59.6	81.3	89.2	**91.6**
tr23	69.3	**90.7**	81.7	85.2
tr31	94.1	93.3	93.9	**94.9**
tr41	94.5	89.6	93.5	**95.7**
tr45	84.7	91.3	91.1	**92.9**
la1	**87.6**	75.2	82.7	87.4
la2	**89.9**	77.3	84.1	88.4
la12	**89.2**	79.4	85.2	89.1
fbis	77.9	73.6	78.0	**80.1**
wap	80.6	68.1	75.1	**81.3**
ohscal	74.6	71.5	62.5	**75.4**
new3	74.4	73.5	67.9	**79.7**

Table 2: The classification accuracy achieved by the different classification algorithms.

Looking at the results of Table 2, we can see that naive Bayesian outperforms the other schemes in five out of the 23 data sets, C4.5 does better in one, the centroid-based scheme does better in 17, whereas the k-nearest-neighbor algorithm never outperforms the other schemes.

A more accurate comparison of the different schemes can be obtained by looking at what extend the performance of a particular scheme is statistically different from that of another scheme. We used two different statistical tests to compare the accuracy results obtained by the different classifiers. The first test is based on the resampled paired t test [13], and the second test is based on the sign test [44]. A brief description of these tests is presented in Appendix A.

The statistical significance results using the resampled paired t test are summarized in Table 3, in which for each pair of classification algorithms, it shows the number of data sets that one performs statistically better, worse, or similarly than the other. Looking at this table, we can see that the centroid-based scheme compared to naive Bayesian, does better in ten data sets, worse in one data set, and they are statistically similar in twelve data sets. Similarly, compared to kNN, it does better in twenty, and it is statistically similar in three data sets. Finally, compared to C4.5, the centroid-based scheme does better in eighteen, worse in one, and statistically similar in four data sets.

The statistical significance results using the sign test are summarized in Table 4, in which for each pair of classifi-

	NB	kNN	C4.5
Cntr	10/1/12	20/0/3	18/1/4
NB		12/4/7	15/3/5
kNN			13/3/7

Table 3: Statistical comparison of different classification algorithms using the resampled paired t test. The entries in the table show the number of data sets that the classifier in the row performs better, worse or similarly than the classifier in the column.

cation algorithms, it shows the z value. The z value was computed based on the average classification accuracy of 10 trials. A z value greater than 1.96, indicates that the classifier of the row is statistically better than the classifier of the column. Looking at this table, we can see that the centroid-based scheme does better than naive Bayesian, kNN, and C4.5. Naive Bayesian does better than C4.5, but does similarly with respect to kNN. Finally, kNN does better than C4.5.

	NB	kNN	C4.5
Cntr	2.71	4.80	4.38
NB		1.46	3.54
kNN			2.71

Table 4: Statistical comparison of different classification algorithms using the sign test. The values in the table are z values and value greater than 1.96 shows that the classifier of the row is statistically better than the classifier of the column.

From these results, we can see that the simple centroid-based classification algorithm outperforms all remaining schemes, with naive Bayesian being second, k-nearest-neighbor being third, and C4.5 being the last. Note that the relative rankings among NB, kNN, and C4.5, agrees to similar results reported in previous works [5, 55, 53, 19].

5 Analysis

The surprisingly good performance of the centroid-based classification scheme suggests that it employs a sound underlying classification model. The goal of this section is to understand this classification model and compare it against those used by other schemes.

In order to understand this model we need to understand the formula used to determine the similarity between a document x, and the centroid vector $\vec{C}$ of a particular class (Equation 8), as this computation is essential in determining the class of x (Equation 9). From Equation 8, we see that the similarity (*i.e.*, cosine) between $\vec{x}$ and $\vec{C}$ is the ratio of the dot-product between $\vec{x}$ and $\vec{C}$ divided by the length of $\vec{C}$. If S is the set of documents used to create $\vec{C}$, then from Equation 6, we have that:

$$\vec{x} \cdot \vec{C} = \vec{x} \cdot \left(\frac{1}{|S|} \sum_{d \in S} \vec{d} \right) = \frac{1}{|S|} \sum_{d \in S} \vec{x} \cdot \vec{d} = \frac{1}{|S|} \sum_{d \in S} \cos(\vec{x}, \vec{d}).$$

That is, the dot-product is the average similarity (as measured by the cosine function) between the new document x and all other documents in the set. The meaning of the length of the centroid vector can also be easily understood using the fact that $\|\vec{C}\|_2 = \sqrt{\vec{C} \cdot \vec{C}}$. Then, from Equation 6 we have that:

$$\|\vec{C}\|_2 = \sqrt{\vec{C} \cdot \vec{C}} = \sqrt{\left(\frac{1}{|S|} \sum_{d \in S} \vec{d} \right) \cdot \left(\frac{1}{|S|} \sum_{d \in S} \vec{d} \right)} = \sqrt{\frac{1}{|S|^2} \sum_{d_i \in S} \sum_{d_j \in S} \vec{d_i} \cdot \vec{d_j}} = \sqrt{\frac{1}{|S|^2} \sum_{d_i \in S} \sum_{d_j \in S} \cos(\vec{d_i}, \vec{d_j})}. \quad (10)$$

Hence, the length of the centroid vector is the square-root of the average pairwise similarity between the documents that support the centroid. There are two things to be noted about this formula; first, this average similarity also includes the self-similarity between the documents in the supporting set; second, because all the documents have been scaled to be of unit length, the length of the centroid vector will always be less or equal to one. In summary, the similarity between a test document and the centroid vector of a particular class, is nothing more than the average similarity between the test document and all the documents in that class, divided by the square-root of the average similarity between the documents in the class itself. (An alternate derivation of the above formulas is presented in [9].)

The above discussion provides us with a qualitative understanding on how the centroid scheme determines the similarity between a test document and a particular class. Essentially, it computes the average similarity between the test document and all the other documents in that class, and then it amplifies that similarity, based on how similar to each other are the documents of that class. If the average pairwise similarity between the documents of the class is small (*i.e.*, the class is *loose*), then that amplification is higher, whereas if the average pairwise similarity is high (*i.e.*, the class is *tight*), then this amplification is smaller.

To better understand this classification model consider the following simple binary classification algorithm, that we will refer to it as $\mathcal{H}$. Let A and B be the two classes, let $\bar{S}_A$ be the average similarity between the items in A, $\bar{S}_B$ be the average similarity between the items in B, and let $\bar{S}_{A,B}$ be the average similarity between all the items (a, b) such that $a \in A$, and $b \in B$. Now consider a test item x, and let $\bar{S}_{x,A}$, and $\bar{S}_{x,B}$ be the average similarities between x and all the items in A and B, respectively. This setting is illustrated in Figure 1. In this classifier, x will be classified as either A or B based on how closely its behavior matches the behavior of the items in class A and the items in class B, as measured by their average similarities.

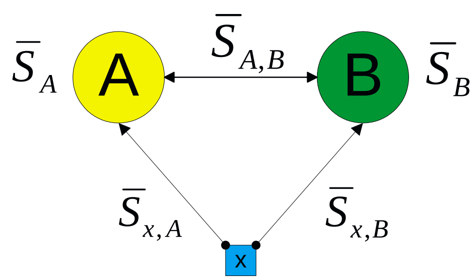

Figure 1: A simple binary classifier.

This behavior can be modeled by looking at the ratios $\bar{S}_A/\bar{S}_{A,B}$ and $\bar{S}_B/\bar{S}_{A,B}$, and comparing them against the ratios $\bar{S}_{x,A}/\bar{S}_{x,B}$ and $\bar{S}_{x,B}/\bar{S}_{x,A}$. The first of these ratios ($\bar{S}_A/\bar{S}_{A,B}$) measures how much *stronger* is the internal similarity between items belonging to class A relative to their similarity to items belonging to class B. Similarly, the second ratio ($\bar{S}_B/\bar{S}_{A,B}$) measures how much stronger is the internal similarity between items belonging to class B relative to their similarity to items belonging to class A. Finally, the last two ratios, measure how much stronger is the similarity of x to the items in A compared to the items in B, and vice-versa. Given the above ratios, then the classification algorithm $\mathcal{H}$ will assign x to class A iff,

$$\frac{\bar{S}_{x,A}/\bar{S}_{x,B}}{\bar{S}_A/\bar{S}_{A,B}} \geq \frac{\bar{S}_{x,B}/\bar{S}_{x,A}}{\bar{S}_B/\bar{S}_{A,B}}, \tag{11}$$

otherwise it will assign in to class B. Essentially, $\mathcal{H}$ compares the strength of the similarity of x to class A relative to the strength of the similarity of items already in A (left side of the inequality), against the strength of the similarity of x to class B relative to the strength of the similarity of items already in B (right side of the inequality), and assigns x to the class for which the relative strength is higher. Performing some simple algebraic manipulations in Equation 11,

and canceling out the $\bar{S}_{A,B}$ terms that appear on both side of the inequality we have that:

$$\frac{\bar{S}_{x,A}/\bar{S}_{x,B}}{\bar{S}_A/\bar{S}_{A,B}} \geq \frac{\bar{S}_{x,B}/\bar{S}_{x,A}}{\bar{S}_B/\bar{S}_{A,B}} \Rightarrow \frac{\bar{S}^2_{x,A}}{\bar{S}_A} \geq \frac{\bar{S}^2_{x,B}}{\bar{S}_B} \Rightarrow \frac{\bar{S}_{x,A}}{\sqrt{\bar{S}_A}} \geq \frac{\bar{S}_{x,B}}{\sqrt{\bar{S}_B}}. \tag{12}$$

We can extend $\mathcal{H}$ to problems with more than two classes, by using a *tournament* method, and thus assigning x to the class for which $\bar{S}_{x,j}/\sqrt{\bar{S}_j}$ is the highest among all classes j.

Now, from the earlier discussion, we know that in the case in which the data items in the above problem are unit-length document vectors, and the similarity is computed using the cosine measure, then from Equation 12 we have that $\mathcal{H}$ will assign x to class A, iff

$$\cos(\vec{x}, \vec{C_A}) \geq \cos(\vec{x}, \vec{C_B}),$$

otherwise x will be assigned to class B; where $\vec{C_A}$ and $\vec{C_B}$ are the centroid vectors of class A and B, respectively. Thus, the classification model used by the centroid-based document classifier is identical to that used by $\mathcal{H}$, that is, it assigns a new document x to the class whose documents better match the behavior of x, as measured by average document similarities.

5.1 Comparison With Other Classifiers

One of the advantages of the centroid-based scheme is that it summarizes the characteristics of each class, in the form of the centroid vector. A similar summarization is also performed by naive Bayesian, in the form of the per-class term-probability distribution functions. Two examples of such centroid vectors for two different collections of documents are shown in Table 5 (these collections are described in Section 4.1). For each of these vectors, Table 5 shows their ten highest weight terms. The number that precedes each term in this table is the weight of that term in the centroid vector. Also note that the terms shown in this table are not the actual words, but their stems.

The advantage of the summarization performed by the centroid vectors is that it combines multiple prevalent features together, even if these features are not simultaneously present in a single document. That is, if we look at the prominent dimensions of the centroid vector (*i.e.*, highest weight terms), these will correspond to terms that appear frequently in the documents of the class, but not necessarily all in the same set of documents. This is particularly important for high dimensional data sets for which the coverage of any individual feature is often quite low. Moreover, in the case of documents, this summarization has the additional benefit of addressing issues related to synonyms, as commonly used synonyms will be represented in the centroid vector. The centroids vectors shown in Table 5 contain various such instances. For example, the tenth centroid of *wap* contains synonym terms like *album* and *record*, the third centroid of *new3* contains synonyms like *japan* and *japanes*, *etc*.. For these reasons, the centroid-based classification algorithm (as well as naive Bayesian) tend to perform better than the C4.5 and the k-nearest neighbor classification algorithms.

The better performance of the centroid-based scheme over the naive Bayesian classifier is due to the method used to compute the similarity between a test document and a class. In the case of naive Bayesian, this is done using Bayes rule, assuming that when conditioned on each class, the occurrence of the different terms is independent. However, this is far from being true in real document collections [27]. One way of understanding the dependence between terms is to look at the degree at which various terms co-occur in the documents of a particular class. If the degree of term co-occurrence is high, then these terms are positively dependent, as the probability of seeing one of the co-occurring terms is high provided that we have seen one of the other co-occurring terms. As the degree of term co-occurrence decreases, the positive dependence also decreases, and after a certain point it gives rise to negative dependence among the terms. In this case, the conditional probability of seeing a certain term is high provided that we have not seen some other terms. The existence of such positive and negative dependence between terms of a particular class causes

naive Bayesian to compute a distorted estimate of the probability that a particular document belongs to that class. If there is positive dependence between the terms in the class, then the probability estimate will be higher than it actually is, whereas if these is negative dependence between the terms, then the probability estimate will be smaller than it actually is. Unfortunately, naive Bayesian has no way by which to account for such term dependence, and much more complicated classifiers such as Bayesian Networks need to be used [16].

On the other hand, the similarity function used by the centroid-based scheme does account for term dependence within each class. From the discussion in Section 5, we know that the similarity of a new document x to a particular class is computed as the ratio of two quantities. The first is the average similarity of x to all the documents in the class, and the second is the square-root of the average similarity of the documents within the class. To a large extent, the first quantity is very similar, in character, to the probability estimate used by the naive Bayesian algorithm, and it suffers from similar over- and under-estimation problems in the case of term dependence. As in the case of naive Bayesian, if the class contains terms that are positively dependent, then the average similarity of x to the documents in the class will be high, as it will tend to match most of the co-occurring terms. Similarly, if the class contains negatively dependent terms, then the average similarity of x to the documents in the class will be small as it will be unnecessarily penalized for not matching the negatively dependent terms.

However, the second quantity of the similarity function, (*i.e.*, the square-root of the average similarity of the documents within the class) does account for term dependency. This average similarity depends on the degree at which terms co-occur in the different documents. In general, if the average similarity between the documents of a class is high, then the documents have a high degree of term co-occurrence (since the similarity between a pair of documents computed by the cosine function, is high when the documents have similar set of terms). On the other hand, as the average similarity between the documents decreases, the degree of term co-occurrence also decreases. Since this average internal similarity is used to amplify the similarity between a test document and the class, this amplification is minimal when there is a large degree of positive dependence among the terms in the class, and increases as the positive dependence decreases. Consequently, this amplification acts as a correction parameter to account for the over- and under-estimation of the similarity that is computed by the first quantity in the document-to-centroid similarity function. We believe that this feature of the centroid-based classification scheme is the reason that it outperforms the naive Bayesian classifier in the experiments shown in Section 4.

This performance difference can be understood in real document data sets. For example, a set of documents containing Clinton-Lewinsky stories will be a more cohesive category than a set of documents containing sports stories such as baseball, football, basketball, and Olympics. In the first category, most of the documents contain words Clinton and Lewinsky and hence these words are frequently co-occurring words. A document tends to belong to this category only if both the words *Clinton* and *Lewinsky* are in the document. On the other hand, any of sports related words like *baseball, football,* and *basketball* appearing in a document will put the document in the second category. Given these two categories, consider a news story containing President Clinton's reaction to the 1995 major league baseball labor dispute between players and owners. This story obviously contains words *Clinton* and *baseball*. The naive Bayesian classifier can easily misclassify this document by assigning to the first category, as the word *Clinton* has a high conditional probability in the first category and *baseball* has relatively lower conditional probability in the second category. However, the centroid-based classifier will most likely classify this document correctly, because the similarity to the first category will be indirectly penalized since the document did not contain the term *Lewinsky*.

6 Discussion & Concluding Remarks

In this paper we focused on a simple linear-time centroid-based document classification algorithm. Our experimental evaluation has shown that the centroid-based classifier consistently and substantially outperforms other classifiers on a wide range of data sets. We have shown that the power of this classifier is due to the function that it uses to compute

the similarity between a test document and the centroid vector of the class. This similarity function can account for both the term similarity between the test document and the documents in the class, as well as for the dependencies between the terms present in these documents.

There are many ways to further improve the performance of this centroid-based classification algorithm. First, in its current form it is not well suited to handle multi-modal classes. However, support for multi-modality can be easily incorporated by using a clustering algorithm to partition the documents of each class into multiple subsets, each potentially corresponding to a different mode [36], or using similar techniques to those used by the generalized instance set classifier [25]. Second, the classification performance can be further improved by using techniques that adjust the importance of the different features in a supervised setting. A variety of such techniques have been developed in the context of k-nearest-neighbor classification [11, 24, 32, 48, 19], all of which can be extended to the centroid-based classifier.

References

[1] M. B. Amin and S. Shekhar. Generalization by neural networks. *Proc. of the 8th Int'l Conf. on Data Eng.*, April 1992.

[2] L. Baker and A. McCallum. Distributional clustering of words for text classification. In *SIGIR-98*, 1998.

[3] D. Boley, M. Gini, R. Gross, E.H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Document categorization and query generation on the world wide web using WebACE. *AI Review (accepted for publication)*, 1999.

[4] D. Boley, M. Gini, R. Gross, E.H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Partitioning-based clustering for web document categorization. *Decision Support Systems (accepted for publication)*, 1999.

[5] W.W. Cohen. Fast effective rule induction. In *Proc. of the Twelfth International Conference on Machine Learning*, 1995.

[6] W.W. Cohen and H. Hirsh. Joins that generalize: Text classification using WHIRL. In *Proc. of the Fourth Int'l Conference on Knowledge Discovery and Data Mining*, 1998.

[7] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[8] T. Curran and P. Thompson. Automatic categorization of statute documents. In *Proc. of the 8th ASIS SIG/CR Classification Research Workshop*, Tucson, Arizona, 1997.

[9] D.R. Cutting, J.O. Pedersen, D.R. Karger, and J.W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the ACM SIGIR*, pages pages 318–329, Copenhagen, 1992.

[10] D.J. Spiegelhalter D. Michie and C.C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

[11] W. Daelemans, S. Gills, and G. Durieux. Learnability and markedness in data-driven acquisition of stress. Technical Report TR 43, Institute for Language Technology and Artificial Intelligence, Tilburg University, Netherlands, 1993.

[12] B.V. Dasarathy. *Nearest neighbor (NN) norms: NN pattern classification techniques*. IEEE Computer Society Press, 1991.

[13] T.G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7), 1998.

[14] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

[15] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[16] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[17] D. E. Goldberg. *Genetic Algorithms in Search, Optimizations and Machine Learning*. Morgan-Kaufman, 1989.

[18] E.H. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. WebACE: A web agent for document categorization and exploartion. In *Proc. of the 2nd International Conference on Autonomous Agents*, May 1998.

[19] Eui-Hong Han. *Text Categorization Using Weight Adjusted* k*-Nearest Neighbor Classification*. PhD thesis, University of Minnesota, October 1999.

[20] W. Hersh, C. Buckley, T.J. Leone, and D. Hickam. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *SIGIR-94*, pages 192–201, 1994.

[21] Makato Iwayama and Takenobu Tokunaga. Cluster-based text categorization: a comparison of category search strategies. In *SIGIR-95*, pages 273–281, 1995.

[22] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proc. of the European Conference on Machine Learning*, 1998.

[23] K. Kira and L.A. Rendell. A practical approach to feature selection. In *Proc. of the 10th International Conference on Machine Learning*, 1992.

[24] I. Kononenko. Estimating attributes: Analysis and extensions of relief. In *Proc. of the 1994 European Conference on Machine Learning*, 1994.

[25] Wai Lam and Chao Yang Ho. Using a generalized instance set for automatic text categorization. In *SIGIR-98*, 1998.

[26] Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *Proc. of the Fifth ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, pages 16–22, 1999.

[27] D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *Tenth European Conference on Machine Learning*, 1998.

[28] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *SIGIR-94*, 1994.

[29] D. Lewis and M. Ringuette. Comparison of two learning algorithms for text categorization. In *Proc. of the Third Annual Symposium on Document Analysis and Information Retrieval*, 1994.

[30] D. D. Lewis. Reuters-21578 text categorization test collection distribution 1.0. *http://www.research.att.com/∼lewis*, 1999.

[31] David D. Lewis, Robert E. Shapire, James P. Callan, and Ron Papka. Training algorithms for linear text classifiers. In *Proceedings of the 19 th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages pages 298–306, 1996.

[32] D.G. Lowe. Similarity metric learning for a variable-kernel classifier. *Neural Computation*, pages 72–85, January 1995.

[33] B. Masand, G. Linoff, and D. Waltz. Classifying news stories using memory based reasoning. In *SIGIR-92*, pages 59–64, 1992.

[34] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[35] Andrew Kachites McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow, 1996.

[36] T.M. Mitchell. *Machine Learning*. WCB/McGraw-Hill, 1997.

[37] J. Moore, E. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, and B. Mobasher. Web page categorization and feature selection using association rule and principal component clustering. In *7th Workshop on Information Technologies and Systems*, Dec. 1997.

[38] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[39] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[40] Forrester Research. Coping with complex data. The Forrester Report, April 1995.

[41] E. Riloff and W. Lehnert. Information extraction as a basis for high-precision text classification. *ACM Transactions on Information Systems*, 12(3), 1994.

[42] J.J. Jr. Rocchio. The SMART retrieval system: Experiments in automatic document processing. In Gerard Salton, editor, *Relevance feedback in information retrieval*. Prentice-Hall, Inc., 1971.

[43] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[44] G.W. Snedecor and W.G. Cochran. *Statistical Methods*. Iowa State University Press, 1989.

[45] TREC. Text REtrieval conference. *http://trec.nist.gov*.

[46] V. Vapnic. *The Nature of Statistical Learning Theory*. Springer, 1995.

[47] S.M. Weiss and C. A. Kulikowski. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufmann, San Mateo, CA, 1991.

[48] D. Wettschereck, D.W. Aha, and T. Mohri. A review and empirical evaluation of feature-weighting methods for a class of lazy learning algorithms. *AI Review*, 11, 1997.

[49] D. Wettschereck and T.G. Dietterich. An experimental comparison of the nearest neighbor and nearest hyperrectangle algorithms. *Machine Learning*, 19:5–28, 1995.

[50] B. Widrow and S.D. Stearns. *Adaptive Signal Processing*. Prentic-Hall, Inc., 1985.

[51] Yahoo! Yahoo! http://www.yahoo.com.

[52] Y. Yang. Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In *SIGIR-94*, 1994.

[53] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval Journal*, May 1999.

[54] Y. Yang and C.G. Chute. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems*, 12(3), 1994.

[55] Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR-99*, 1999.

[56] Y. Yang and J. Pederson. A comparative study on feature selection in text categorization. In *Proc. of the Fourteenth International Conference on Machine Learning*, 1997.

A Measures of Statistical Significance

The Resampled *t* Test One way of measuring the statistical difference between the performance of two classification algorithms is to use the resampled paired t test [13]. This test compares the performance of two classification algorithms based on the results from n trials. In each trial, data set is randomly divided into a training set and and a test set. The error rates of algorithms A and B on the test set are recorded. Let $p_A^{(i)}$ be the error rate of algorithm A and $p_B^{(i)}$ be the error rate of algorithm B during trial i. Then Student's t test can be computed using the statistic:

$$t = \frac{\bar{p}\sqrt{n}}{\sqrt{\frac{\sum_{i=1}^{n}(p^{(i)}-\bar{p})^2}{n-1}}},$$

where $p^{(i)} = p_A^{(i)} - p_B^{(i)}$ and $\bar{p} = \frac{1}{n}\sum_{i=1}^{n} p^{(i)}$. This statistic has a t distribution with $n-1$ degrees of freedom. For 10 trials used in the experiments reported in Section 4.2, the null hypothesis that two classifiers are not different in terms of performance can be rejected if $|t| > t_{9,0.975} = 2.262$.

The Signed Test Another statistical test that can be used to compare different classification algorithm is the sign test [44]. Given n data sets, let n_A be the number of data sets that classifier A does better than classifier B in terms of the classification accuracy. Then we have

$$\frac{\frac{n_A}{n} - p}{\sqrt{\frac{p \times q}{n}}} \approx N(0, 1)$$

where p is the probability that classifier A does better than classifier B; and $q = 1 - p$. Under the null hypothesis, $p = 0.5$, so

$$z = \frac{\frac{n_A}{n} - 0.5}{\sqrt{\frac{0.5 \times 0.5}{n}}} \approx N(0, 1)$$

We can reject the null hypothesis that two classifiers are the same in terms of performance if $|z| > Z_{0.975} = 1.96$.

	wap									
1	0.20 diana	0.17 film	0.13 showbiz	0.13 notabl	0.13 angel	0.13 annual	0.12 albert	0.12 lo	0.12 award	0.12 festiv
2	0.26 emmi	0.23 cb	0.22 tv	0.21 rate	0.21 nbc	0.20 adult	0.16 abc	0.14 household	0.13 program	0.12 fox
3	0.19 studi	0.19 research	0.19 cell	0.18 risk	0.18 cancer	0.16 patient	0.15 diseas	0.14 women	0.13 heart	0.12 drug
4	0.41 newspap	0.22 editor	0.19 advertis	0.14 media	0.13 peruvian	0.13 coverag	0.12 percent	0.12 journalist	0.12 press	0.12 circul
5	0.25 exhibit	0.21 auction	0.21 stolen	0.20 art	0.18 gogh	0.16 draw	0.16 sculptor	0.15 paint	0.14 galleri	0.13 van
6	0.38 film	0.19 box	0.16 million	0.15 star	0.14 offic	0.13 weekend	0.13 festiv	0.13 pictur	0.12 top	0.12 movie
7	0.33 stock	0.21 dow	0.18 compani	0.17 percent	0.14 greenspan	0.14 industri	0.14 busi	0.14 financi	0.13 wire	0.13 pr
8	0.49 cable	0.21 network	0.15 fcc	0.15 rate	0.14 usa	0.13 showtim	0.13 hbo	0.12 espn	0.12 channel	0.11 deal
9	0.34 week	0.34 bestsell	0.26 weekli	0.25 publish	0.22 hardcov	0.19 paperback	0.19 book	0.13 nea	0.10 fo	0.10 morton
10	0.29 album	0.28 music	0.23 record	0.23 song	0.14 band	0.13 concert	0.12 sold	0.12 rock	0.11 stone	0.10 diana
11	0.39 clinton	0.27 senat	0.27 house	0.24 white	0.23 campaign	0.20 reform	0.19 republican	0.15 financ	0.13 vote	0.13 presid
12	0.27 game	0.17 smith	0.15 coach	0.14 season	0.13 win	0.13 championship	0.12 se	0.11 nomo	0.11 player	0.11 marlin
13	0.14 charact	0.13 film	0.11 david	0.11 music	0.11 product	0.10 review	0.09 michael	0.09 sound	0.09 john	0.08 costum
14	0.33 internet	0.25 microsoft	0.22 comput	0.19 zdnet	0.19 wir	0.15 access	0.15 servic	0.15 reserv	0.14 technologi	0.14 compani
15	0.37 ticket	0.28 hottest	0.28 opera	0.24 theater	0.19 broadwai	0.19 receipt	0.16 lyric	0.13 week	0.13 net	0.12 funk
16	0.36 casino	0.34 farm	0.27 legion	0.20 trump	0.20 mirag	0.18 miami	0.18 aid	0.16 concert	0.13 wow	0.12 deauvill
17	0.43 internet	0.35 onlin	0.24 comput	0.18 servic	0.17 microsoft	0.16 web	0.14 america	0.13 compuserv	0.13 site	0.13 compani
18	0.28 murdoch	0.16 disnei	0.15 compani	0.15 stock	0.15 usa	0.13 network	0.13 viacom	0.12 million	0.12 seagram	0.12 stake
19	0.28 daili	0.22 hollywood	0.21 insid	0.20 front	0.18 fox	0.17 tv	0.16 film	0.14 ink	0.12 deal	0.11 pictur
20	0.48 dvd	0.24 game	0.23 player	0.21 toshiba	0.15 emeri	0.13 typ	0.12 video	0.11 digit	0.11 compact	0.10 alien

	new3									
1	0.34 waste	0.29 dump	0.26 water	0.26 pollution	0.23 sea	0.22 environment	0.20 river	0.18 radioact	0.16 nuclear	0.14 russia
2	0.44 export	0.37 cocom	0.22 russian	0.18 control	0.18 technologi	0.16 russia	0.13 missil	0.12 german	0.11 arm	0.11 dual
3	0.52 japan	0.35 japanes	0.23 tokyo	0.18 trade	0.14 insur	0.14 talk	0.13 kyodo	0.12 market	0.12 framework	0.12 auto
4	0.41 nuclear	0.41 korea	0.31 north	0.30 iaea	0.25 korean	0.18 dprk	0.17 inspect	0.14 pyongyang	0.12 seoul	0.12 pakistan
5	0.41 al	0.28 palestinian	0.24 israe	0.20 arab	0.20 lebanon	0.19 hizballah	0.17 israel	0.15 abu	0.14 terrorist	0.14 hama
6	0.34 grain	0.32 agricultur	0.20 price	0.19 rice	0.18 product	0.16 percent	0.14 farm	0.14 market	0.14 farmer	0.14 rural
7	0.37 newspap	0.26 publish	0.23 press	0.17 media	0.16 public	0.15 editor	0.13 russian	0.12 magazin	0.12 book	0.12 print
8	0.29 murder	0.18 al	0.16 kill	0.14 polic	0.12 terrorist	0.11 assassin	0.11 crime	0.10 court	0.10 death	0.10 people
9	0.52 nuclear	0.26 ukrain	0.21 korea	0.20 iaea	0.19 treati	0.16 north	0.16 dprk	0.14 weapon	0.14 korean	0.13 prolifer
10	0.55 drug	0.24 traffick	0.23 gang	0.23 polic	0.20 heroin	0.17 arrest	0.16 narcot	0.16 kg	0.15 addict	0.12 cocain
11	0.49 nafta	0.40 mexico	0.24 job	0.23 mexican	0.17 american	0.15 trade	0.15 worker	0.13 export	0.11 agreem	0.11 wage
12	0.60 violenc	0.40 women	0.26 domest	0.17 crime	0.16 abus	0.15 speaker	0.15 victim	0.14 batter	0.12 bill	0.11 prevent
13	0.33 china	0.23 trade	0.22 embargo	0.22 mfn	0.18 clinton	0.16 right	0.16 vietnam	0.14 human	0.12 haiti	0.11 polici
14	0.56 earthquak	0.24 quake	0.22 insur	0.21 disast	0.15 california	0.15 volcano	0.14 dollar	0.12 reinsur	0.11 speaker	0.11 amend
15	0.48 submarin	0.32 rosyth	0.26 trident	0.23 devonport	0.21 defenc	0.19 nuclear	0.18 dockyard	0.16 refit	0.15 vsel	0.14 missil
16	0.44 pulp	0.41 paper	0.30 price	0.24 cent	0.22 mill	0.17 newsprint	0.13 compani	0.13 cdollar	0.12 profit	0.11 cost
17	0.61 tax	0.29 pound	0.28 cent	0.22 vate	0.19 incom	0.18 rate	0.12 taxe	0.10 taxat	0.09 budget	0.09 uk
18	0.44 drug	0.30 traffick	0.28 cocain	0.26 cartel	0.17 colombian	0.16 colombia	0.15 cali	0.14 polic	0.13 mafia	0.12 crime
19	0.36 speci	0.25 whale	0.23 endang	0.23 wolve	0.22 wildlif	0.17 hyph	0.17 blank	0.16 mammal	0.15 marin	0.15 wolf
20	0.30 rwanda	0.25 rebel	0.24 africa	0.17 kill	0.17 hutu	0.17 kigali	0.16 unita	0.16 tutsi	0.15 african	0.15 rwandan
21	0.38 project	0.31 dam	0.24 hydroelectr	0.21 power	0.19 hyph	0.18 electr	0.15 gorge	0.15 hydropow	0.15 river	0.13 construct
22	0.53 vw	0.36 lopez	0.29 gm	0.24 opel	0.21 volkswagen	0.21 piech	0.19 motor	0.14 espionag	0.12 german	0.10 compani
23	0.14 hous	0.13 pound	0.13 properti	0.13 home	0.12 liv	0.12 house	0.12 retir	0.12 life	0.11 people	0.11 social
24	0.35 fuel	0.32 energi	0.31 plutonium	0.27 nuclear	0.24 reactor	0.19 electr	0.17 power	0.14 coal	0.13 cell	0.13 japan
25	0.54 women	0.23 parti	0.22 elect	0.21 labour	0.16 vote	0.16 parliam	0.15 candid	0.14 mp	0.12 seate	0.11 democr
26	0.56 argentina	0.33 argentin	0.31 falkland	0.23 bueno	0.23 aire	0.17 tella	0.17 malvina	0.16 british	0.15 island	0.12 skyhawk
27	0.59 bank	0.25 imf	0.23 world	0.15 lend	0.12 develop	0.11 loan	0.11 project	0.11 monetari	0.11 dollar	0.11 preston
28	0.29 tax	0.23 helmslei	0.22 hunter	0.18 ir	0.18 evasion	0.17 fraud	0.15 dominelli	0.14 rose	0.13 guilti	0.12 feder
29	0.39 polic	0.30 kill	0.23 policeman	0.21 offic	0.14 policemen	0.13 murder	0.13 milit	0.12 shot	0.11 bomb	0.11 asyut
30	0.42 school	0.38 educ	0.38 curriculum	0.32 teacher	0.26 test	0.19 patten	0.19 pupil	0.13 teach	0.12 old	0.10 ron
31	0.42 tunnel	0.29 rail	0.25 eurotunnel	0.22 channel	0.17 freight	0.16 ferri	0.15 kent	0.15 pound	0.13 br	0.12 railwai
32	0.45 journalist	0.20 hostag	0.20 kong	0.19 hong	0.11 lebanon	0.11 arrest	0.11 kill	0.10 releas	0.10 china	0.10 polic
33	0.32 spratli	0.31 vietnam	0.19 sea	0.19 island	0.18 territori	0.17 china	0.16 russian	0.15 vietnames	0.14 disput	0.14 oil
34	0.64 drug	0.20 legal	0.16 greif	0.15 court	0.14 colombia	0.14 addict	0.13 de	0.11 traffick	0.11 bogota	0.11 decrimin
35	0.24 boate	0.23 ship	0.19 piraci	0.18 vessel	0.16 kong	0.15 hong	0.14 pirat	0.14 hijack	0.14 sea	0.13 fish
36	0.37 food	0.32 hyph	0.27 fda	0.25 label	0.18 blank	0.18 fsi	0.17 poultri	0.16 drug	0.15 cfr	0.15 addit
37	0.38 nobel	0.36 prize	0.15 peace	0.11 soviet	0.11 award	0.11 gorbachev	0.10 walesa	0.09 mandela	0.09 menchu	0.09 dalai
38	0.34 drug	0.30 prozac	0.23 lilli	0.19 sale	0.18 pharmaceut	0.18 cent	0.17 patient	0.16 depress	0.13 merck	0.13 solvai
39	0.36 iraq	0.30 matrix	0.27 inquiri	0.27 churchill	0.25 scot	0.18 export	0.16 lord	0.16 defenc	0.12 tool	0.11 sir
40	0.35 azt	0.34 drug	0.30 patient	0.27 amgen	0.27 aid	0.25 epo	0.16 hiv	0.15 wellcom	0.14 infect	0.13 diseas
41	0.40 pharmaceut	0.34 drug	0.25 cent	0.24 compani	0.20 glaxo	0.19 research	0.17 pound	0.14 amp	0.14 sale	0.13 dollar
42	0.47 tourism	0.45 tourist	0.27 visitor	0.17 hotel	0.16 cent	0.14 percent	0.09 cuba	0.09 increas	0.08 attract	0.08 million
43	0.32 soviet	0.31 nato	0.26 cfe	0.21 europ	0.20 treati	0.18 tank	0.17 arm	0.16 convent	0.16 bush	0.15 gorbachev
44	0.43 forest	0.41 amazon	0.26 brazil	0.18 mende	0.17 brazilian	0.16 environment	0.13 ecuador	0.12 deforest	0.12 rain	0.11 rio

Table 5: The ten highest weight terms in the centroids of two data sets.